# CHAPTER 1 – The Role of Algorithms in Computing

## 1.1 – Algorithms

### 1.1-1

A real-world example that requires sorting could be something like sorting emails by alphabetical order in some way or another, over date and time.

A real-world example of finding the shortest distance between two points could be a GPS or something of the sort, where you want to find the fastest way between two points.

### 1.1-2

You probably want to consider resource usage alongside speed.

### 1.1-3

A data structure I have seen would be arrays. Arrays are strong in the way that they are easy to pull data from at any point, insert data at a point or even remove it. But they can be limited in a way that they are less resource efficient than say, a single linked list in the instance of wanting to traverse and print all the elements in them.

### 1.1-4

They both want to have the most efficient path. However, the shortest path does not necessarily include all points. A salesperson would want to, given that there is value in going to those locations, but the shortest path does not need to be inclusive of that desire.

### 1.1-5

The best solution is necessary in situations such as deciding optimal numbers to use when it comes to machines to maximize runtime efficiency and the lowest use of storage. It saves money to do this. But any situation may do when dealing with people because they are not static like computers. They have uncontrollable variables which means that whenever they are given an optimal solution, they will likely deviate from it.

### 1.1-6

I think a good example of this is having a job where people are scheduled to tasks on a daily basis. You assume that everyone will be there on time so you have information on how you can utilize your employees. However, sometimes those employees don’t arrive on time, so an alternate solution may arise when you learn that information.

## 1.2 – Algorithms as a technology

### 1.2-1

Email interfaces constantly use algorithms at the application level to resort mail into different categories all the time.

### 1.2-2

8n^2 < 64nlgn

n < 8lgn

n/8 < lgn

2^(n/8) < n

2 is always the lower bound and this inequality breaks at 44 by brute force, so;

2 <= n <= 43

### 1.2-3

100n^2 < 2^n

10n < 2^(n/2)

1 is always the lower bound and this inequality breaks at 15 by brute force, so;

2 <= n <= 14

## CHAPTER PROBLEMS

### 1-1

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1  second | 1  minute | 1  hour | 1  day | 1  month | 1  year | 1  century |
|  |  |  |  |  | \*2^30 | \*2^12 | \*2^10 |
|  |  | \*60^2 | \*60^2 | \*24^2 | \*30^2 | \*12^2 | \*10^2 |
|  |  | \*60 | \*60 | \*24 | \*30 | \*12 | \*10 |
|  | Not sure how to solve this at the moment |  |  |  |  |  |  |
|  |  | \*sqrt(60) | \*sqrt(60) | \*sqrt(24) | \*sqrt(30) | \*sqrt(12) | \*sqrt(10) |
|  |  | \*cbrt(60) | \*cbrt(60) | \*cbrt(24) | \*cbrt(30) | \*cbrt(12) | \*cbrt(10) |
|  |  | \*log2(60) | \*log2(60) | \*log2(24) | \*log2(30) | \*log2(12) | \*log2(10) |
|  |  | Inverse factorial stuff… | … | … | … | … | … |